**Back-end functionalities**

We choose SpringBoot framework to develop our back-end microservice because of it’s high-integration with other technologies. Many technology features are adopted. Specifically, we use “MySQL Connector” to connect to the MySQL database, “Spring AOP” to print the logs, “Spring IoC” to manage various Java Beans, “Spring-swagger” to perform back-end self-test. “fastjson” to convert JSON string, and “Github-pageHelper” to implement page-division stategy. Back-end functionalities would be explained in the context of these technologies.

1. **Environment change**

We use different environments to speed up the development of our web application (Spookify), including dev, sit, uat and prd. These environments could be easily changed during application start up with the help of SpringBoot.

1. **Log service**

We create our log service by using Spring AOP. Annotation (SpookifyInfo) is added on each controller to weave the log content into the normal controller call. SpookifyInfo could show certain controller is being called, the calling process ends and the exceptions encountered during the calling process.

1. **Data persistence**

All the sensitive data is stored in the back-end MySQL database. The data of Contact Me section is stored in *item* table. customer data is stored in the *customer* table, and visitor data is stored in the *visitor* table. Front-end could fetch and store data easily by invoking different back-end controllers.

1. **Highly customizable data storage layer**

We use MyBatis and MyBatis-Plus to flexibly manipulate business data. When requirements change or new requirements derive, back-end interface could be changed accordingly by modifying the mapper provided by MyBatis.

The customizable of data could also be reflected in the tables itself. Every table has “data” field, which stores JSON string. Front-end developers could store any business data within this field.

1. **Back-end data validation**

Although most of the data verification work is done on the front end, key data is still verified on the back end. The main purpose of back-end data validation is to ensure the safety of sensitive data stored in the database. Another purpose of this strategy is to avoid users directly access the back-end microservice. When user registers, the back-end service layer will call the dao layer interface to check whether there are duplicate user names and return the result to the front-end.For another example, when user inserts data into the *item* table, the back-end will verify whether there is a matching user id (user id is a foreign key in *item* table).

The back-end not only verifies data entry into the database, but also verifies whether the parameters passed in by the front-end are compliant. This verification is performed on the relevant annotations on the controller. For example, NotNull annotation indicates whether the field is empty while Pattern annotations indicates if the field match certain regular expression.

By using these back-end data validation mechanics, the burden of front-end could be relatively reduced. Front-end could concentrate on complex field validation.

1. **Record status tracking**

Back-end MySQL database preserves the status of each record. Whenever user performs CRUD operations, back-end database would store the timestamp and the type of the operation. Therefore, every record is traceable. Front-end developer could examine this information to perform data validation or business logic.

1. **Integrated back-end self-test tool**

By importing Swagger package and configuring the relative fields, back-end developers can test each controller using webpage. Swagger is more like a GUI tool, visualizing the input data model, the returned data model, the request methods and request URL of different controllers. Usage of Swagger not only benefits back-end developers, but also benefits front-end developers. Because in Spookify, front-end developers need to frequently use Swagger to identify the information of various interfaces.

1. **Paging**

When the number of records exceed the maximum capacity of one page, paging strategy should be implemented. The implementation could be as simple as using the “limit” keyword in SQL. However, “Github Page Helper” provides broader functionalities. Users could specify the amount of records in one page, the total number of pages and the additional requirements. Also, by using Github Page Helper. Front-end data could be easily parsed and delivered to back-end with safety.

1. **Page redirection**

Page redirection is also implemented in the context of controller. In SpringBoot, the return value of each controller would be verified and identified. SpringBoot would justify whether the value (ModelandView) is physical or logic. If it is a ModelandView, then controller would redirect requests to logic ModelandView (In our project, it is JSP). Front-end developers could switch the current web page by calling the target controller.

1. **Token and Session**

Token and session are the common methods for web-based application to save the personal preference of user. Session id is saved in the web application. After user logging in, cookie id would be sent to server, and server generated a session id. By generating session ids and stored them in database. Users could skip the log in part as well as keep their personal preference.

1. **QR code generator**

Hutool tool kit allow us to generate QR code in back-end. By passing the genereated QR code to the front-end with the help of controller, front-end could parse data and place the QR code on our login page.

In summary, the back-end of Spookify is a SpringBoot based web application. Spookify uses MySQL to achieve data durability.